AEC Computing and Applied Mathematics Center

AEC RESEARCH AND DEVELOPMENT REPORT

TID-4500
14th Ed.

NYO-2879
PHYSICS

COMBINATORIAL PROPERTIES OF
MACHINE SHOP SCHEDULING

by

Jack Heller

July 1, 1959

Institute of Mathematical Sciences

NEW YORK UNIVERSITY

NEW YORK, NEW YORK

AEC Computing and Applied Mathematics Center
Institute of Mathematical Sciences
New York University

TID-4500
14th Ed.

NYO-2879
PHYSICS

COMBINATORIAL PROPERTIES OF MACHINE SHOP SCHEDULING

by
Jack Heller

July 1, 1959

NYO-2879
PHYSICS AND
MATHEMATICS

ABSTRACT

Combinatorial properties of deterministic machine shop scheduling is studied.

Expressions for the idle and schedule times are obtained for any schedule. Estimates for the number of different schedules and schedule times are obtained. The question of consistent schedules is discussed and methods for the generation of consistent schedules are exhibited.

Various objective functions for the determination of best schedules are obtained.

TABLE OF CONTENTS

COMBINATORIAL PROPERTIES OF MACHINE SHOP SCHEDULING

1. Introduction.

Combinatorial properties of the deterministic machine shop scheduling [8] for a finite number of jobs is studied.

In particular we discuss the explicit rules (Section 2 and Appendix) in order to fix the area of study (deterministic machine shop scheduling) clearly in our minds. We are able to obtain explicit expressions for the idle and schedule times in terms of the processing times and ordering of the objects through the machines.

Estimates of the number of different schedules and the number of different schedule times are obtained. We find two estimates for the number of different schedule times: one which depends on the processing times and one which is purely combinatorial as it does not depend on the processing times but only on the number of jobs and machines. A weak estimate for the number of different schedules is obtained and indicates, as in [3], the fact that there are more schedules than schedule times for many given sets of processing times.

We investigate the question of consistent schedules and find a result in our notation that is similar to other investigators [1,5]. The stress on consistency treated here

affords us a simple means of obtaining consistent randomly generated schedules. We discuss some methods of generating consistent schedules which are closely related to the generation of latin rectangles.[1/]

Various criteria for determining "best" schedules is formulated in our notation.

2. A Machine Shop Model.

In a previous paper [3] an explicit formulation of the rule of a scheduling model was given. We will adhere to this model and note heuristically the one change which requires the study presented in this paper.

Roughly speaking we want to discuss some properties of the schedule time as it depends on the order in which objects (or jobs) are routed through machines. In order to follow out this program we must note the properties of the order relations and have a notation which allows us to express and deduce properties of the order relations. These properties are purely combinatorial. Furthermore, we must have properties of the processing times and a notation which allow us to express and deduce properties of the processing times.

The properties which we will adhere to are given in [3]. These properties express the intuitively clear facts

---

[1/] B. Giffler and G. L. Thompson give examples of arrays connected with scheduling problems in their report, "Algorithms for Solving Production Scheduling Problems," (June 1959 - Draft Copy, IBM Research Center).

(i) only one object (job) can be performed on a machine at a time, (ii) each object must pass through the machines in a prescribed manner, (iii) the processing time t_{mj} of each object j on each machine m does not depend on the order in which the objects are processed, and (iv) each object is processed as soon as possible subject only to the order requirements on each machine and the order requirements for each object. In [3] the order requirements of each object were the same, i.e. each object proceeded from machine 1 to machine 2, ... to machine M, the last machine. (This order is that found in flow shops and assembly lines.) The only difference between the present formulation and the formulation in [3] is that we allow each object to have any order of processing. Presumably the order of processing objects is given.

This model of a machine shop is quite simple: it assumes one machine at each station and that the processing time of the jobs is deterministic, i.e. does not vary in a known stochastic manner due to unforeseen happenings.

To keep track of the order relations of the jobs we use the symbols $\preccurlyeq$ and (mj). The symbol (mj) stands for the phrase the j^{th} job on the m^{th} machine, a phrase that we will use over and over again. The binary relation $\preccurlyeq$ relates the symbols (mj), i.e. the order relation between two job processes on either the same job or on two different jobs. Intuitively the binary relation $\preccurlyeq$ means

is started before or at the same time as. Thus $(13) \precsim (36)$ means the job 3 on the 1^{st} machine is started before or at the same time as job 6 on the third machine.

If two objects are different and are related by $\precsim$, we can replace $\precsim$ by $\prec$, which we take to mean is started before.

The numbering of the machines and jobs is arbitrary (i.e. preconceived) and does not in general correspond to the order in which the jobs are processed. It will be necessary for us to consider permutations of the job order on a particular machine, permutations of the machine order for a particular job and even permutations of both the machine order and job order. We designate an element, i.e. a job designation respectively as (mj_k), $(m_\ell j)$ and $(m_\ell j_k)$.

The various sets of orderings on each machine for a given schedule, i.e. specification of all orderings of all objects through the machines, will be designated for each machine as $P_m \mathcal{M}_m = \{(mj_k) | k=1,\ldots,J\}$. These sets, one for each machine, are called the scheduled ordering [7]. If the scheduled ordering on a particular machine is the preconceived ordering, we write $\mathcal{M}_m = \{(mj) | j=1,2,\ldots,J\}$. Thus, we are using P_m to designate a job permutation on the m^{th} machine.

The various sets of orderings for each job, i.e. specification of all orderings of each object through the machines will be designated as $Q_j \mathcal{J}_j = \{(m_\ell j) | \ell=1,\ldots,M\}$.

These sets, one for each job, are called the technological ordering [7]. If the technological ordering for a particular job is the preconceived ordering, we write $\mathcal{J}_j = \{(mj) | m=1,\ldots,M\}$. Thus, we are using Q_j to designate a machine permutation for the j^{th} job.

In the sets $P_m\mathcal{M}_m$ and $Q_j\mathcal{J}_j$ the order of appearance of the elements (mj) will be taken as the order of processing the objects. Thus for each $P_m\mathcal{M}_m$, $(mj_1) \prec (mj_2) \prec \ldots \prec (mj_J)$ and for each $Q_j\mathcal{J}_j$, $(m_1 j) \prec (m_2 j) \prec \ldots \prec (mj_J)$. (These ordering properties are said to be simply ordered [2].) We have been able to use the binary relation $\prec$ which means <u>is started before</u> in each of the sets $P_m\mathcal{M}_m$ and $Q_j\mathcal{J}_j$ because each job in the machine ordering $P_m\mathcal{M}_m$, i.e. scheduled ordering, <u>is a different job</u> and each machine in the job ordering $Q_j\mathcal{J}_j$, i.e. technological ordering, <u>is a different machine</u>.

It is convenient to call all the job designations in a given technological ordering <u>equivalent</u>. We designate equivalence by the binary relation $\simeq$. Thus $(m_3 j) \simeq (m_7 j)$ means the j^{th} job on the m_3 machine is equivalent to the j^{th} job on the m_7 machine. Equivalence intuitively means, in our context, is the same job.

These assumptions about the ordering properties of the jobs, of course, restrict us to a special type of machine shop. We do not allow an object to be put on the same

machine more than once. (If an object is not put on a particular machine, we can attach a 0 processing time and still give the object a position in technological and scheduled orderings without effecting idle and schedule times. This fact will be evident from the theorems below.)

A schedule is a collection of technological orderings $\{Q_j \mathcal{J}_j\}$ and a collection of scheduled orderings $\{P_m \mathcal{M}_m\}$. Since the elements appearing in $\{Q_j \mathcal{J}_j\}$ are the same as the elements appearing in $\{P_m \mathcal{M}_m\}$, the order relations given by the $P_m \mathcal{M}_m$'s may be inconsistent with the order relations given by the $Q_j \mathcal{J}_j$'s. Many investigators have studied this question of consistency; we mention but two -- Akers and Friedman [1] and Marimont [5] to indicate two approaches to this problem. We will give a third approach which is more in keeping with the philosophy of statistical sampling (i.e. simulation) techniques of determining properties of consistent schedules. A consistent schedule will be designated as $\{P_m \mathcal{M}_m\} \circledast \{Q_j \mathcal{J}_j\}$.

By a "best" schedule we mean the schedule (or schedules) that gives a minimum to a numerically defined function of each schedule. These numerically defined functions are of various types: the due date type, the in process inventory cost type, the minimum time type, the minimum idle time type, ... Whichever type it may be, we need to have expressions for the idle time and the schedule time. These times depend

on the order of processing the objects and the processing times of the objects. We have made special assumptions about the processing times t_{mj} ; and therefore we study a special type of machine shop.

In deriving expressions for the idle and schedule times we must assume that each object is processed as soon as possible subject only to the consistent schedule order relations. By this statement we mean that an object must be processed on a given machine when (i) it has already been processed on all the previous machines that are required by this object's technological ordering $Q_j \mathcal{J}_j$ and when (ii) the machine it is to be processed on has processed all the objects that are required to be processed before this object's scheduled ordering $P_m \mathcal{M}_m$.

From these assumptions a typical Gantt diagram would be as given in Figure 1 and corresponds to the simply ordered sets

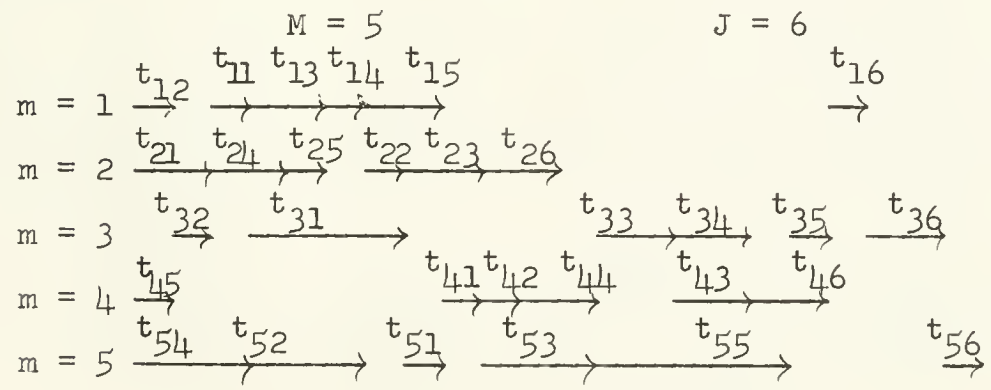

Figure 1. A Typical Gantt Diagram for a Machine Shop Schedule.

$Q_1 \mathcal{J}_1 = \{(21)(11)(31)(51)(41)\}$; $m_1=2$, $m_2=1$, $m_3=3$, $m_4=5$, $m_5=4$

$Q_2 \mathcal{J}_2 = \{(12)(32)(52)(22)(42)\}$; 1 3 5 2 4

$Q_3 \mathcal{J}_3 = \{(13)(23)(53)(33)(43)\}$, 1 2 5 3 4

$Q_4 \mathcal{J}_4 = \{(54)(24)(14)(44)(34)\}$; 5 2 1 4 3

$Q_5 \mathcal{J}_5 = \{(45)(25)(15)(55)(35)\}$; 4 2 1 5 3

$Q_6 \mathcal{J}_6 = \{(26)(46)(16)(36)(56)\}$; $m_1=2$, $m_2=4$, $m_3=1$, $m_4=3$, $m_5=5$

and

$P_1 \mathcal{M}_1 = \{(12)(11)(13)(14)(15)(16)\}$; $j_1=2$, $j_2=1$, $j_3=3$, $j_4=4$, $j_5=5$, $j_6=6$

$P_2 \mathcal{M}_2 = \{(21)(24)(25)(22)(23)(26)\}$; 1 4 5 2 3 6

$P_3 \mathcal{M}_3 = \{(32)(31)(33)(34)(35)(36)\}$; 2 1 3 4 5 6

$P_4 \mathcal{M}_4 = \{(45)(41)(42)(44)(43)(46)\}$; 5 1 2 4 3 6

$P_5 \mathcal{M}_5 = \{(54)(52)(51)(53)(55)(56)\}$; $j_1=4$, $j_2=2$, $j_3=1$, $j_4=3$, $j_5=5$, $j_6=6$

There is a closely related rectangular display of the elements of $P_m \mathcal{M}_m$ and $Q_j \mathcal{J}_j$, such that each row contains the elements of each $P_m \mathcal{M}_m$ and each column contains the elements of each $Q_j \mathcal{J}_j$ such that no j is the same in the columns and all order relations given by $\{P_m \mathcal{M}_m\}$ and $\{Q_j \mathcal{J}_j\}$ are satisfied from left to right. For the Gantt diagram given in Figure 1 we have the rectangular display

(12) (11) (13) (14) (15) (16)

(21) (24) (25) (22) (23) (26)

(32) (31) (33) (34) (35) (36)

(45) (41) (42) (44) (43) (46)

(54) (52) (51) (53) (55) (56).

This display is closely connected with latin rectangles.

In the Appendix we give a formal development of the above heuristic ideas. It is possible to skip the Appendix and understand the remaining sections which contain results of use in applications, provided the meaning of the symbols $\precsim$, $\prec$, (mj), $(m_\ell j)$, (mj_k), $P_j \mathcal{M}_j$, $Q_j \mathcal{J}_j$ and $\{P_m \mathcal{M}_m\} \textcircled{x} \{Q_j \mathcal{J}_j\}$ are understood. The purpose of the finicky formulation of the scheduling rules is to indicate how scheduling and sequencing problems can be brought within a known and highly developed mathematical discipline. If this can be done and if this is done, the results and tools of the mathematical discipline can be brought to bear on the problem on hand. This procedure should be one of the aims of "Operations Research."

3. The Idle Time and Schedule Time.

In many of the problems connected with machine shop scheduling it is necessary to know the schedule time as a function of the processing times [8]. We will proceed to develop the concepts that will lead to an expression for the schedule time.

We define the idle time before processing the (mj_k) and after processing the (mj_{k-1}), which we denote as I_{mj_k}, as the time the m^{th} machine remains idle. The required idleness of the m^{th} machine depends on the

processing times of the jobs and the order relations given by the consistent[2] schedule $\{P_m \mathcal{M}_m\} \otimes \{Q_j \mathcal{J}_j\}$. We define the schedule time, which we denote as $t(P)$, as the total time to process all the jobs. The schedule is symbolically designated by P.

We will have many occasions to consider the time to finish processing a particular job on a particular machine. Thus we give a definition for this concept:

The time to finish processing the (mj_k) under the schedule $\{P_m \mathcal{M}_m\} \otimes \{Q_j \mathcal{J}_j\}$ will be designated as $t(P|mj_k)$. We will refer to $t(P|mj_k)$ as the job finish time.

The hazy concept of a bottleneck or being held up is related to the abstract concept of covering. This concept is central to the discussion of idle time; and hence, we give [cf. ref. 2] the definition of covering: by "$(m'j')$ is covered by (mj)" we mean that $(m'j') \prec (mj)$ and $\nexists$ a $(m''j'') \ni (m'j') \prec (m''j'') \prec (mj)$ for a given schedule. We will write $\subset$ for "is covered by."

Intuitively "is covered by" means "may possibly hold up" for if $(m'j') \subset (mj)$ then $(m'j') \prec (mj)$ and hence $(m'j')$ may hold up the processing of (mj). It may happen that $(m'j')$ may not hold up the processing of (mj) for some other $(m''j'')$ may hold up the processing of the (mj). This question of which $(m'j')$ holds up which (mj)

2/ For the remainder of this paper $\{P_m \mathcal{M}_m\} \otimes \{Q_j \mathcal{J}_j\}$ will denote a consistent schedule unless explicitly stated otherwise.

depends on the schedule and processing times.

We prove

Lemma 1.1. For a given schedule, if $(mj') \subset (mj_k)$, then $(mj') = (mj_{k-1})$; if $(m'j) \subset (m_\ell j)$, then $(m'j) = (m_{\ell-1} j)$.

Proof: In the set $P_m \mathcal{M}_m$ we have $(mj_{k-1}) \prec (mj_k) \not\simeq (mj_{k-1})$. We have $(mj_{k-1}) \in Q_{j_{k-1}} \mathcal{J}_{j_{k-1}}$ and only one set $Q_{j'} \mathcal{J}_{j'}$. We call $(mj_{k-1}) = (m_\ell j)$ and have $(mj_{k-1}) \prec (m_{\ell+1} j)$. Since $(m_\ell j) \simeq (mj_{k-1})$ and $(m_\ell j) \not\simeq (mj_k)$, we see that $(m_\ell j)$ is not related via the binary relation $\prec$ to (mj_k). Hence there are no other elements related to (mj_k) other than those $(m_{\ell'} j) \prec (m_\ell j) \simeq (mj_{k-1})$ and those $(mj_{k'}) \prec (mj_k)$. Thus we have $(mj_{k-1}) \subset (mj_k)$ and the first part of the lemma is proved. The second part is proved in a similar fashion.

The diagrams [2] of partially-ordered sets are closely connected to the Gantt diagrams. If a circle is drawn around the beginning of the arrows in a Gantt diagram and if those circles are connected which are at the beginning of arrows representing a pair of elements (mj) and $(m'j')$ such that $(mj) \subset (m'j')$, we have a diagram of a partially ordered set laid on its side. In Figure 2 we have the diagram on its side corresponding to the Gantt diagram given in Figure 1. We note

Lemma 2.1. Every element (mj) is covered by two, one, or no elements. If two elements cover (mj), one of the

covering elements $\in P_m \mathcal{M}_m$ and one of the covering elements is in $Q_j \mathcal{J}_j$.

The proof is quite simple for Lemma 1.1 tells us that if an element is covered by two elements, one covering comes from the elements of one of the sets $P_m \mathcal{M}_m$ and another comes from the elements of one of the sets $Q_j \mathcal{J}_j$. If an element is both a last element of a set $P_m \mathcal{M}_m$ and $Q_j \mathcal{J}_j$ it is not covered. If an element is a last element of a set $P_m \mathcal{M}_m$ but not a last element of $Q_j \mathcal{J}_j$ it is covered by an element of $Q_j \mathcal{J}_j$ only and similarly for the case of an element last in $Q_j \mathcal{J}_j$ and not last in $P_m \mathcal{M}_m$.

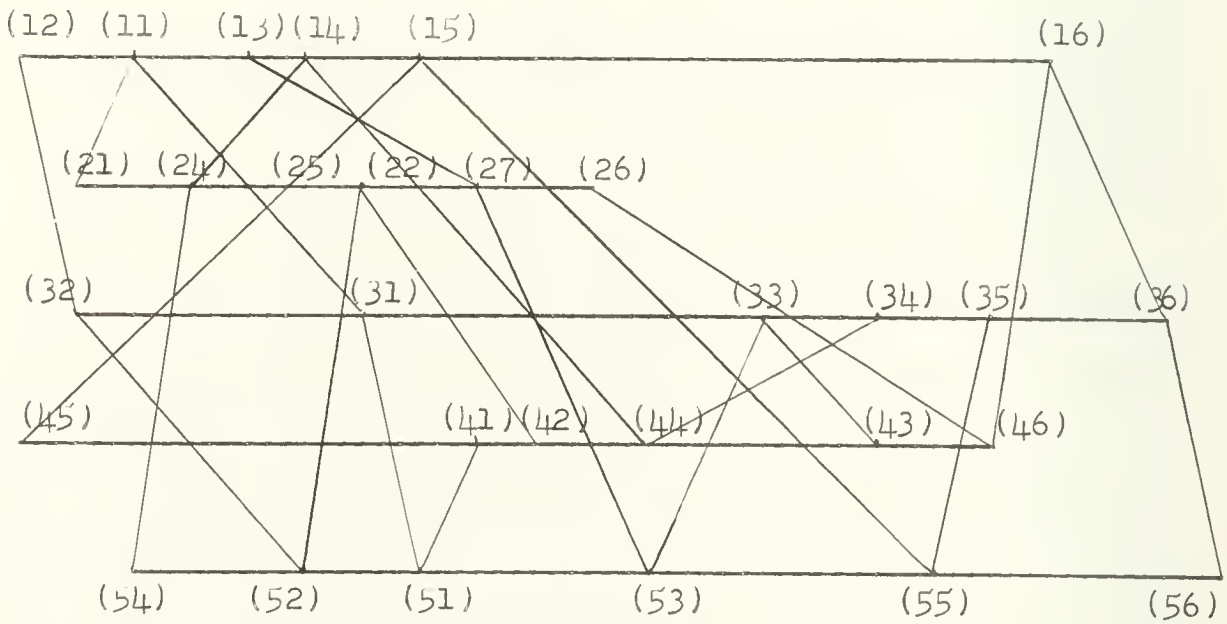

Figure 2. The diagram of the partially ordered set turned on its side corresponding to the Gantt Diagram given in Figure 1.

We need the following definition before writing an expression for the idle time. By $t(P|m'j' \backsim mj_k)$ we mean the time to finish processing the $(m'j') \backsim (mj_k)$ under the schedule $\{P_m M_m\} \textcircled{x} \{a_j \beta_j\}$.

Idle time comes about when a machine ready to receive an object for processing cannot process this object because it is being processed on another machine. The time the m^{th} machine finishes processing the j_{k-1} object is $t(P|mj_k, \backsim mj_k)$. The time to finish processing the object to be processed on the m^{th} machine as the j_k^{th} object is $t(P|m_\ell, j \backsim mj_k)$ [3]. If $t(P|m_\ell, j \backsim mj_k) - t(P|mj_k, \backsim mj_k) > 0$ we have idle time and if the expression is negative we do not have idle time as follows from (A_2) and the definitions for idle job finish times. We can use the maximum notation to express

Lemma 3.1. The idle time before processing the (mj_k) is

$$(3.1) \quad I_{mj_k} = \max \begin{pmatrix} t(P|m_\ell, j \backsim mj_k) - t(P|mj_k, \backsim mj_k) \\ 0 \end{pmatrix} .$$

From the definitions of the job finish times and covering we have

$$(3.2) \qquad t(P|mj_k, \backsim mj_k) = t(P|mj_{k-1}) .$$

---

[3] In $t(P|(m_\ell, j \backsim mj_k)$ we understand that $m_\ell \neq m$. Hence $(m_\ell, j) \simeq (mj_k)$.

However, the time to finish processing the (mj_{k-1}) is

$$(3.3) \qquad t(P|mj_{k-1}) = \sum_{k'=1}^{k-1} (t_{mj_{k'}} + I_{mj_{k'}})$$

which follows from (A5) and Definitions 4 and 8 in the Appendix. Combining (3.1-3), we have

$$(3.4) \quad t(P|mj_k) = t(P|mj_{k-1}) + t_{mj_k} + I_{mj_k}$$

$$= \max \left\{ \begin{array}{l} t(P|m_{\ell',j} \subset mj_k) + t_{mj_k} \\ \\ t(P|mj_{k-1}) + t_{mj_k} \end{array} \right\} .$$

We can solve (3.4) for $t(P|mj_k)$ and eventually obtain an expression for the schedule time in terms of the processing times t_{mj} and the schedule $\{P_m m_m\} \textcircled{x} \{Q_j \, \mathcal{J}_j\}$. Firstly, we note from (3.4) that

$$(3.5) \qquad t(P|mj_1) = t(P|m_{\ell',j} \subset mj_1) + t_{mj_1} ;$$

secondly, from (3.4) and (3.5) we have

$$(3.6) \quad t(P|mj_2) = \max \left\{ \begin{array}{l} t(P|m_{\ell',j} \subset mj_2) + t_{mj_2} \\ \\ t(P|m_{\ell',j} \subset mj_1) + t_{mj_1} + t_{mj_2} \end{array} \right\} .$$

Continuing on in this manner, we have

Lemma 4.1. The time to process the (mj_k) is

$$(3.7) \quad t(P|mj_k) = \max_{1 \leq s \leq k} \left\{ t(P|m_{\ell',j} \subset mj_s) + \sum_{k'=s}^{k} t_{mj_{k'}} \right\} .$$

The schedule time is given by

Lemma 5.1. The schedule time is

$$(3.8)\quad t(P) = \max_{m} \max_{1\le s\le J} \left\{ t(P|m_{\ell'} j \frown m j_s) + \sum_{k'=s}^{J} t_{m j_{k'}} \right\}.$$

We now proceed to solve (3.7) for the time $t(P|mj_J)$ as a function of the processing times t_{mj} and the schedule $\{P_m \mathcal{M}_m\} \circledast \{Q_j \mathcal{J}_j\}$. In (3.7) with $k = J$ let k_L be that value of s for which the maximum of the terms behind the maximum operator is reached:

$$(3.9)\quad t(P|mj_J) = t(P|m_{\ell'} j \frown m j_{k_L}) + \sum_{k'=k_L}^{J} t_{m j_{k'}}.$$

Since the first term in (3.9) will include idle as well as processing times and the second term contains only processing times, it follows from (A5) that j_{k_L} is that j_{k_L} for which $I_{m j_{k_L}} \neq 0$ and $I_{m j_{k_L+1}} = \ldots = I_{m j_J} = 0$.

If we call $(m_{\ell'} j) \frown (m j_{k_L})$ $(m_{L-1} j_{K_{L-1}})$, we can rewrite (3.9) as

$$(3.10)\quad t(P|mj_J) = t(P|m_{L-1} j_{K_{L-1}}) + \sum_{k'=k'_L}^{J} t_{m j_{k'}}.$$

Using (3.7), we obtain

$$(3.11)\quad t(P|mj_J) = \max_{1\le s\le K_{L-1}} \left\{ t(P|m_{\ell'} j \frown m_{L-1} j_s) + \sum_{k'=s}^{K_{L-1}} t_{m_{L-1} j_{k'}} \right\} + \sum_{k'=k_L}^{J} t_{m j_{k'}}.$$

Continuing on as before, we have

$$(3.12)\quad t(P|mj_J) = t(P|m_{L-2}j_{k_{L-2}}) + \sum_{k'=k_{L-1}}^{K_{L-1}} t_{m_{L-1}j_{k'}} + \sum_{k'=k_L}^{J} t_{mj_{k'}},$$

where $j_{k_{L-1}}$ is that $j_{k_{L-1}}$ for which $I_{mj_{k_{L-1}}} \neq 0$ and $I_{mj_{k_{L-1}+1}} = \ldots = I_{mj_{K_{L-1}}} = 0$. Finally, we arrive at

<u>Theorem</u> 1. <u>The time to finish processing all the objects on machine</u> m <u>under the schedule</u> $\{P_m \mathcal{M}_m\} \circledast \{Q_j \mathcal{J}_j\}$ <u>is</u>

$$(3.13)\quad t(P|m_L j_J) = \sum_{\ell'=L}^{1} \sum_{k'=k_{\ell'}}^{K_{\ell'}} t_{m_{\ell'}j_{k'}}$$

<u>where</u> $k_1 = 1$ <u>and</u> $K_L = m$. <u>The integers</u> k_ℓ <u>and</u> K_ℓ <u>are members of the set</u>

$$(3.14)\quad \mathcal{K}_m = \Big\{k_\ell, K_\ell \,\Big|\, (m_\ell j_{K_\ell}) \prec (m_{\ell+1} j_{k_{\ell+1}}) \ni I_{m_{\ell+1}j_{k_{\ell+1}}} \neq 0$$

$$\text{and } I_{m_{\ell+1}j_{k_{\ell+1}+1}} = \ldots = I_{m_{\ell+1}j_{K_{\ell+1}}} = 0\Big\}$$

There is a simple Gantt diagram interpretation of the result (3.13). In Figure 1 if $m_L = 3$, the expression (3.13) becomes

$$\begin{aligned} t(P|3j_6) = {} & t_{36} \\ & + t_{16} \\ & + t_{46} + t_{43} \\ & + t_{33} \\ & + t_{53} \\ & + t_{23} + t_{22} \\ & + t_{52} + t_{54}, \end{aligned}$$

where $L = 7$ and

$$\begin{aligned} m_7 &= 3 & K_7 &= 6 & k_7 &= 6 \\ m_6 &= 1 & K_6 &= 6 & k_6 &= 6 \\ m_5 &= 4 & K_5 &= 6 & k_5 &= 5 \\ m_4 &= 3 & K_4 &= 3 & k_4 &= 3 \\ m_3 &= 5 & K_3 &= 4 & k_3 &= 4 \\ m_2 &= 2 & K_2 &= 5 & k_2 &= 4 \\ m_1 &= 5 & K_1 &= 2 & k_1 &= 1 \, . \end{aligned}$$

This path through the Gantt diagram starts out at $(m_L j_{K_L}) = (36)$ moves back along the machine 3 until there is idle time before one of the objects. Then we switch to the machine whose object covers the above mentioned object on machine 3 which had idle time. We contine, as above, to work back until we finally come to an object that is processed at the beginning of the schedule, i.e. does not cover any other object.

From the definition of the schedule time we have

Corollary 1.1. The schedule time

$$t(P) = \max_{\mathcal{K}_m} \sum_{\ell'=L}^{1} \sum_{k'=k_\ell}^{K} t_{m_{\ell'} j_{k'}} \, . \tag{3.15}$$

In many problems a best schedule is defined to be that schedule which gives a minimum schedule time. In our notation we have

Corollary 1.2. The minimum schedule time

$$t_{min} = \min_{P} \max_{\mathcal{K}_m} \sum_{\ell'=L}^{1} \sum_{k'=k_\ell}^{K} t_{m_{\ell'} j_{k'}} \, . \tag{3.16}$$

If the processing times t_{mj} are integers, we have

Corollary 1.3. If the processing times are integers, the schedule times are integers.

The elements (mj) which correspond to the strings given by k_ℓ and K_ℓ of $\mathcal{K}_m$ form an interesting system. Let us define the set of (mj) which appear in the sums of (3.15) as $\mathcal{L}_m$:

$$(3.17)\quad \mathcal{L}_m = \{(mj) \mid (mj) = (m_{\ell'}, j_{k'}),\ \ell'=1,\dots,L;\ k'=k_{\ell'},\dots,K_{\ell'};\ \Rightarrow k_{\ell'}, K_{\ell'} \in \mathcal{K}_M\}.$$

For each $(mj) \in \mathcal{L}_m$, $(mj) \preceq (mj)$. If we have two distinct elements (mj) and $(m'j') \in \mathcal{L}_m$, the elements are either in the same $P_m\mathcal{M}_m$, if $m = m'$, or the elements are in two distinct $P_m\mathcal{M}_m$, if $m \neq m'$. If the distinct elements (mj) and $(m'j') \in P_m\mathcal{M}_m$, either $(mj) \prec (m'j')$ or $(m'j') \prec (mj)$. If the distinct elements (mj) and $(m'j') \in \mathcal{L}_m$ are in different $P_m\mathcal{M}_m$'s, it is still true that either $(mj) \prec (m'j')$ or $(m'j') \prec (mj)$ because the last element of one of the $P_m\mathcal{M}_m$ is covered by the first element of the other $P_m\mathcal{M}_m$. In a similar fashion we have the transitive relation between any three elements $(mj) \in \mathcal{L}_m$ and if $(mj) \preceq (m'j')$ and if $(m'j') \preceq (mj)$ for any two elements in $\mathcal{L}_m$, we have $(mj) = (m'j')$. We have thus proved

Theorem 2. The elements $(mj) \in \mathcal{L}_m$ are simply ordered under the binary relation $\preceq$.

We will give one more result before proceeding to the study of bounds on the number of different schedules and schedule times. The one result is an expression for the idle time which is convenient for numerical calculation.

From the expression for the idle time given by Lemma 3.1 and (3.3) we obtain

$$(3.18)\quad I_{mj_k} = \max \left\{ \begin{array}{c} t(P|m_{\ell'}j \frown mj_k) - \sum_{k'=1}^{k-1} (t_{mj_{k'}} + I_{mj_{k'}}) \\ 0 \end{array} \right\}.$$

The first term behind the maximum operator expresses the time to complete the object equivalent to (mj_k) before being processed on the m^{th} machine. Let us call $(mj_k) \in P_m \mathcal{M}_m$ which is in the set $Q_j \mathcal{J}_j \quad (m_\ell j)$. Then

$$(3.19)\quad t(P|m_{\ell'}j \frown mj_k) = \sum_{\substack{j' \ni \\ (m_{\ell-1}j_{k'}) \prec (m_\ell j)}} (t_{m_{\ell-1}j_{k'}} + I_{m_{\ell-1}j_{k'}}).$$

Since all other sums of the form $\sum_{\substack{j' \ni \\ (m_{\ell'}j') \prec (m_\ell j)}} (t_{m_{\ell'}j'} + I_{m_{\ell'}j'})$ where $(m_{\ell'}j') \in Q_j \mathcal{J}_j$ are smaller than the sum given in (3.19), because $(m_{\ell'}j') \prec (m_{\ell-1}j)$ and $(m_{\ell'}j')$ is not covered by $(m_\ell j)$, we can write

$$(3.20)\quad t(P|m_{\ell'}j \frown mj_k) = \max_{m' \ni} \sum_{\substack{j_{k'} \ni \\ (m'j) \prec (mj_k) \simeq (m'j) \quad (m'j_{k'}) \prec (mj_k)}} (t_{m'j_{k'}} + I_{m'j_{k'}}).$$

Combining (3.20) and (3.18), we obtain

Theorem 3. The idle time before processing the (mj_k)

$$(3.21)\quad I_{mj_k} = \max\left\{ \begin{array}{l} \max\limits_{\substack{m' \\ (m'j) \prec (mj_k) \simeq (m'j)}} \left\{ \sum\limits_{j_{k'} \rightarrow} (t_{m'j_{k'}} + I_{m'j_{k'}}) - \sum\limits_{k'=1}^{k-1} (t_{mj_{k'}} + I_{mj_{k'}}) \right\} \\ \qquad (mj_{k'}) \prec (mj_k) \\ 0 \end{array} \right\}$$

If all the objects are processed in the same order on all the machines and we number the machines $1,2,\ldots,M$, the $\max\limits_{m'}$ term in (3.21) is largest for $m' = m-1$. We thus have the case of the assembly line schedule [3]:

Corollary 3.1. For the assembly line schedule the idle time before processing the (mj_k) is given by

$$\max\left\{ \begin{array}{l} \sum\limits_{\substack{k' \rightarrow \\ (m-1j_{k'}) \prec (mj_k)}} (t_{m-1j_{k'}} + I_{m-1j_{k'}}) - \sum\limits_{k'=1}^{k-1} (t_{mj_{k'}} + I_{mj_{k'}}) \\ 0 \end{array} \right\}.$$

The expression for the idle time given in (3.21) is useful because of its recursive nature.

4. Estimate of the Number of Different Schedules and the Number of Different Schedule Times.

There are two viewpoints from which we are able to easily obtain estimates for the number of different schedules and the number of different schedule times.

One method makes use of the fact that if the processing times are integers the schedule time is also an integer [cf. Corollary 1.3]; the other method is purely combinatorial in essence and depends on the form of the schedule time given by Corollary 1.1. Because of the importance of these results from the statistical sampling point of view, we proceed to obtain these estimates.

Because of the simple ordering of the objects processed through each machine (A1-2) and the independence of processing time on ordering (A4), we have

Lemma 6.3. The minimum schedule time τ_{min} satisfies the inequality

$$\max_{m'} \sum_{j'=1}^{J} t_{m'j'} \leq \tau_{min} . \tag{4.1}$$

If we do not say anything about the technological orderings expressed by $\{Q_j\}$, we can only give the following crude estimate to the maximum schedule time τ_{max} :

Lemma 7.3. The maximum schedule time τ_{max} satisfies the inequality

$$\tau_{max} \leq \sum_{m'=1}^{M} \sum_{j'=1}^{J} t_{m'j'} , \tag{4.2}$$

for the largest schedule time will be obtained, if possible, when there is no simultaneous operation.

If the processing times are integers, the schedule

times are also integers [cf. Lemma 1.3]. Hence, we have from Lemmas 6.3 and 7.3

Theorem 4. If the processing times are integers, the number of different processing times $N(t(P))$ satisfies the inequality

$$(4.3) \qquad N(t(P)) \leq \sum_{m'=1}^{M} \sum_{j'=1}^{J} t_{m'j'} - \max_{m'} \sum_{j'=1}^{J} t_{m'j'+1} .$$

(When the processing times are not integers but rational numbers, we can always multiply these processing times by a suitable constant so that they are integers.)

A second estimate to the number of different schedule times can be obtained from the expression for the schedule time given by Theorem 1. The least number of terms, for any schedule, that can appear in the sums of (3.15) is J: this is the case of one machine having no idle time and taking the longest time to process all the objects. The most number of terms that can appear in the sums of (3.15) is MJ: this is the case when there is no simultaneous operation. If all the processing times are different, an upper bound to the number of different schedule times will be the number of different sums possible in (3.15). We thus have

Theorem 5. The number of different schedule times satisfies the inequality

$$(4.4) \qquad N(t(P)) \leq \sum_{n=J}^{M \cdot J} \binom{M \cdot J}{n} .$$

We now turn our attention to the estimates of the number of different consistent schedules. In general the number of different schedules will depend on the technological orderings given by the $\{Q_j\}$. As this problem is very difficult and not of general use for our purposes, we will only obtain a simple estimate for the number of different schedules.

We can choose the order of processing the objects on one of the machines arbitrarily. Objects that must be processed before the order assumed on this machine as required by the technological orderings $\{Q_j \mathcal{J}_j\}$ can be chosen arbitrarily only subject to the order requirements given by $\{Q_j \mathcal{J}_j\}$. Since there are $J!$ possible orderings on the machine with arbitrarily chosen order, we have the weak

Lemma 8. The number of different orderings, $N(P)$, satisfies the inequality

$$N(P) \geq J! \quad . \tag{4.5}$$

Although the above result is rather weak, we can still draw the conclusion from Theorem 4 and Lemma 7.3 that in many cases the number of different schedules is far greater than the number of different schedule times [cf. ref. 3 for specific examples]. We are thus led to the realization that sampling techniques may be of value in obtaining "good" schedules.

5. Techniques for Random Generation of Consistent Schedules.

Statistical sampling techniques for the determination of "good" schedules requires that each schedule in the sample be consistent. We could generate schedules arbitrarily and if they are consistent use them and if they are not consistent reject them; or we could, if possible, find a method of generating consistent schedules and use all schedules generated. The first method is costly in computing, even though there are techniques for determining consistency [5], and so we will concentrate our efforts on the second method.

In the case of assembly line scheduling [3], the question of consistency never appeared. The reason for the non-appearance of consistency in the assembly line scheduling is given by

Lemma 9. (Assembly Line Scheduling). If the technological ordering is the same for all jobs, i.e. all $Q_j \mathcal{J}_j = \mathcal{J}_j$, then any schedule ordering, i.e. any $\{P_m \mathcal{M}_m\}$ is consistent;

Lemma 10. If the schedule ordering is the same for all jobs, i.e. all $P_m \mathcal{M}_m = \mathcal{M}_m$, then any technological ordering, i.e. any $\{Q_j \mathcal{J}_j\}$ is consistent.

The proofs are simple and follow from Theorem 2, and Rules (A 1-2).

We will consider the generation of random consistent machine shop schedules when the $\{Q_j \mathcal{J}_j\}$ are given. Thus, we must determine the $\{P_m \mathcal{M}_m\}$ such that the schedule

$\{P_m \mathcal{M}_m\} \otimes \{Q_j \mathcal{J}_j\}$ is consistent. There are many ways of accomplishing this desired end of which we will consider only two.

Let us consider a rectangular array such that the first row contains elements $(1j_k)$ and blanks, the second row contains elements $(2j_k)$ and blanks, ... For each set of elements $(mj) \in Q_j \mathcal{J}_j$ the order relations given by $Q_j \mathcal{J}_j$ are adhered to in the sense that if $(m_\ell j)$ and $(m_{\ell'} j) \in Q_j \mathcal{J}_j$ and if $(m_\ell j) \prec (m_{\ell'} j)$ then $(m_\ell j)$ will appear in a column that is to the left of the column containing $(m_{\ell'} j)$. A schedule formed from this array by choosing the $\{P_m \mathcal{M}_m\}$ for each row as the elements (mj_k) appear from left to right is consistent, because the $\{Q_j \mathcal{J}_j\}$ are ordered as described above and no element is in a loop [cf. Theorem 2].

As an example of generating the rectangular array, let us consider the $\{Q_j \mathcal{J}_j\}$ used in the example of Figure 1. We will consider the sets in the order $Q_1 \mathcal{J}_1$, $Q_2 \mathcal{J}_2$, ..., $Q_6 \mathcal{J}_6$. We place the elements of $Q_1 \mathcal{J}_1$ in their proper rows (as described above) such that the first element in $Q_1 \mathcal{J}_1$ appears in column one, the second element in $Q_1 \mathcal{J}_1$ appears in column two, ... :

(11)

(21)

(31)

(41)

(51)

We now choose elements from $Q_2 \mathcal{J}_2$ and place them in columns in the same manner as the elements of $Q_1 \mathcal{J}_1$ were placed, except if there is already an element occuplying a space, we move the element of $Q_2 \mathcal{J}_2$ over to the right to the next available column:

(12) (11)
(21) (22)
(32) (31)
(41) (42)
(52) (51)

Continuing on in this manner, we obtain the rectangular array

(12) (11) (13) (14) (15) (16)
(21) (24) (25) (22) (23) (26)
(32) (31) (33) (34) (35) (36)
(45) (41) (42) (44) (43) (46)
(54) (52) (51) (53) (55) (56) .

The $\{P_m \mathcal{M}_m\}$ are read off from the rows, e.g.

$$P_1 \mathcal{M}_1 = \{(12)\ (11)\ (13)\ (14)\ (15)\ (16)\} .$$

All the sets $P_m \mathcal{M}_m$ are as given in the example of Section 2.

If we choose the $\{Q_j \mathcal{J}_j\}$ randomly, e.g. $Q_2 \mathcal{J}_2$, $Q_1 \mathcal{J}_1$, $Q_3 \mathcal{J}_3$, $Q_6 \mathcal{J}_6$, $Q_5 \mathcal{J}_5$ and $Q_4 \mathcal{J}_4$ and form the above rectangular array for these sets, we would obtain a

randomly generated consistent $\{P_m \mathcal{M}_m\}$. Not all possible consistent schedules will be obtained in this manner, for conflicts in the rectangular array between two elements were resolved by placing the second chosen element of the two conflicting elements to the right of the first element. If, however, we randomly determine which element is to come first whenever a conflict occurs and if we adhere to the column ordering rule, we would obtain all possible consistent orderings.

For example in the above case, i.e. $Q_1 \mathcal{J}_1$, $Q_2 \mathcal{J}_2$, $Q_3 \mathcal{J}_3$, $Q_4 \mathcal{J}_4$, $Q_5 \mathcal{J}_5$, $Q_6 \mathcal{J}_6$, if we resolved the conflict between elements (41) and (42) by choosing $(42) \prec (41)$ than we would obtain the array

(12) (11) (13) (14) (15) (16)
(21) (24) (25) (22) (23) (26)
(32) (31) (33) (34) (35) (36)
(45) (42) (41) (44) (43) (46)
(54) (52) (51) (53) (55) (56).

In this case all $P_m \mathcal{M}_m$ are the same as before except

$$P_4 \mathcal{M}_4 = \{(45)\ (42)\ (41)\ (44)\ (43)\ (46)\}.$$

If all conflicts were resolved randomly we would generally obtain different $P_m \mathcal{M}_m$ for the same choice of $Q_1 \mathcal{J}_1$, $Q_2 \mathcal{J}_2$, ..., $Q_6 \mathcal{J}_6$.

A second method of generating consistent schedules is

to work with columns in the rectangular array. First we choose some order for $\{Q_j \mathcal{J}_j\} = \{Q_{j_1}\mathcal{J}_{j_1}, \cdots, Q_{j_J}\mathcal{J}_{j_J}\}$. Then we place the first element of the set $Q_{j_1}\mathcal{J}_{j_1}$ in its proper row in column 1. Next we choose the first element of the set $Q_{j_2}\mathcal{J}_{j_2}$ and place it into the first column if there is no conflict or randomly determine which element is to be to the right. We continue on in this fashion until all elements are placed in the rectangular array. Then we read off the $\{P_m \mathcal{M}_m\}$, and obtain a consistent schedule $\{P_m \mathcal{M}_m\} \circledast \{Q_j \mathcal{J}_j\}$.

Other methods of generating consistent schedules can be formed by randomly combining the above two techniques.

Once a schedule is formed, it is an easy computation to evaluate the schedule time $t(P)$ using Theorem 5, Lemma 8.3 and (3.3). Statistical sampling techniques to find the minimum schedule time are practical and the solution of the statistical decision problem previously posed [3] applies here. To complete the analysis of this statistical decision problem we need only know the distribution of schedule times over the set of feasible schedules. We will consider this problem in a subsequent paper.4/

4/ The analysis of this problem shows as in [3] that the schedule times are asymptotically normally distributed over the set of schedules. This property is quite general for constrained objective functions and the statistical sampling technique can be used. Any special requirements about the objective function and constraints, such as convexity for the statistical sampling approach are unnecessary. However, a difficulty appears in generating a random sample consistent with the constraints. These results will appear elsewhere.

6. Various Optimization Criteria.

It is not always desirable to minimize the schedule time to determine a "best" schedule [4,8]. We will discuss other optimization criteria and exhibit various forms of the objective functions.

The due date criteria is one of the considerations for a best schedule: we designate the time that an object is desired from the start of processing as D_j.

If we attach a penalty to an object that is not finished when it is desired, we can construct objective functions which can be used to determine a "best" schedule. For example, if the penalty of not completing the j^{th} object per unit time is c_j, then the penalty under a given schedule for the j^{th} object is, say,

$$Z_j = c_j \max\left\{ \begin{array}{c} t(P|m_M j) - D_j \\ \\ 0 \end{array} \right\} . \tag{6.1}$$

If the j^{th} object is finished before its due date, we pay no penalty for $t(P|m_M j)-D_j < 0$; if the j^{th} object is finished after its due date, we pay the penalty $c_j\{t(P|m_M j)-D_j\}$.

There are two objective functions other than the schedule time commonly used to determine a best schedule [8]:

(i) minimization of the total penalty of not completing an object on time, and

(ii) minimization of the maximum penalty of not completing an object on time.

In case (i) the objective function is

$$Z_1 = \sum_{j=1}^{J} c_j \max \left\{ \begin{matrix} t(P|m_M j)-D_j \\ \\ 0 \end{matrix} \right\} \tag{6.2}$$

and a best schedule is one which gives $\min_P Z_1$. In case (ii) the objective function is

$$Z_2 = \max_j \left\{ c_j \max \left\{ \begin{matrix} t(P|m_M j)-D_j \\ \\ 0 \end{matrix} \right\} \right\}, \tag{6.3}$$

and a best schedule is one which gives $\min_P Z_2$.

It is easily seen that these objective functions can be written in general as $\max \{\text{function } [t(P|m_M j), t(P|m_M J), D_1, \ldots, D_J]\}$ and the best schedule has the form of a minimum of a maximum, bringing out the game theoretic aspects of the schedule choice. In any event it seems reasonable that statistical sampling will be of use in choosing "good" schedules in the deterministic machine shop scheduling.

APPENDIX

The Rules of a Particular Machine Shop Scheduling.

In order to pursue the intricacies of the combinatorial properties of machine shop scheduling we will have to carefully define and explicitly state the rules satisfied by the symbols we use. We proceed to define these symbols and state these rules or assumptions.

During the course of our discussion, we will have many occasions to use the phrase "the j^{th} object on the m^{th} machine" where "object" and "machine" are assumed known as we all know them. The numbering of the objects (or jobs) and machines is arbitrary and preconceived. If we have occasion to consider a permutation of the objects with respect to the preconceived order, we will designate the objects as $j_1, j_2, \ldots, j_k, \ldots, j_J$ where J is the total number of objects considered; if we have occasion to consider a permutation of the machines with respect to the preconceived order, we will designate the machines as $m_1, m_2, m, \ldots, m_M$ were M is the total number of machines considered. Thus, we are led to

Definition 1. The phrase "the j^{th} object on the m^{th} machine" will be symbolized by (mj), where the numbering of jobs and machines is arbitrary and preconceived. For a permutation of the objects and machines with respect to the preconceived ordering the phrase "the j_k^{th} object on the m_ℓ^{th}

machine" is symbolized by $(m_{\ell} j_k)$.

In all sequencing problems we must deal with order relations between the objects processed on each machine and the order relations for each object processed through the machines.[5] We are therefore led to the defining of a binary relation:

Definition 2. The binary relation $\preccurlyeq$ between any two elements, e.g. $(mj) \preccurlyeq (m'j')$, will be taken to mean intuitively (mj) is processed before or at the same time as $(m'j')$.

For each machine we have to consider the ordering of the distinct objects on each machine; i.e. (mj_k), $k=1,\ldots,J$, with respect to the binary relation $\preccurlyeq$. If we call the sets[6] $\{(mj) \mid j=1,\ldots,J\} = \mathcal{M}_m$, $m=1,2,\ldots,M$, we require[7]

(A1) All $\mathcal{M}_m$ and all permutations of the elements of $\mathcal{M}_m$ designated as $P_m \mathcal{M}_m$ are simply ordered,[8] i.e.

(i) For all $(mj_k) \in P_m \mathcal{M}_m$ $(mj_k) \preccurlyeq (mj_k)$;

(ii) If $(mj_k) \preccurlyeq (mj_{k'})$
and $(mj_{k'}) \preccurlyeq (mj_k)$,
then $(mj_k) = (mj_{k'})$.

5/ Nelson and Jackson [7] call these orderings the scheduled ordering and technological ordering respectively.

6/ We will use the braces $\{\ \}$ to indicate a set.

7/ For a detailed discussion of the reasons for these rules for assembly line scheduling see [3].

8/ We will use the definitions given by Birkhoff [2] for the various types of orderings that we will have occasion to use.

(iii) If $(mj_k) \prec (mj_{k'})$
and $(mj_{k'}) \prec (mj_{k''})$,
then $(mj_k) \prec (mj_{k''})$.

(iv) For all $(mj_k) \neq (mj_{k'})$
either $(mj_k) \prec (mj_{k'})$
or $(mj_{k'}) \prec (mj_k)$.

There is an equivalence relation between the elements of the sets $\mathcal{M}_m$ which relates the same job on different machines. We define this equivalence relation via Definition 3. The elements of the $\{\mathcal{M}_m\}$ are said to be equivalent if they refer to the same job. Using the binary relation $\simeq$ to mean equivalent, we have for all $(mj) \in \{\mathcal{M}_m\}$

(i) All $(mj) \simeq (mj)$.

(ii) If $(mj) \simeq (m'j)$
and $(m'j) \simeq (m''j)$
then $(mj) \simeq (m''j)$, and

(iii) If $(mj) \simeq (m'j)$
then $(m'j) \simeq (mj)$.

By definition the elements of each set $\mathcal{M}_m$ refer to different jobs. Hence, we have

Lemma A 1. Any two distinct elements in $\mathcal{M}_m$ are not equivalent.

To specify the order in which each job is processed

through the machines (technological ordering), we consider the maximum, disjoint, equivalence sets of $\{\mathcal{M}_m\}$. Calling these sets $\mathcal{J}_j$, we explicitly define them via

Definition 4. The sets $\mathcal{J}_j$ are constructed from the sets $\mathcal{M}_m$ ⇒

(i) All $(mj) \in \mathcal{J}_j$ are equivalent

(ii) $\mathcal{J}_j \cap \mathcal{J}_{J_{j'}} = \emptyset$ (the null set) $j \neq j'$, and

(iii) $\bigcup_j \mathcal{J}_j = \bigcup_m \mathcal{M}_m$.

We construct the sets $\mathcal{J}_j$ in the following manner: (i) from $P_1\mathcal{M}_1$ choose (mj_1), (ii) find all the elements equivalent to (mj_1) in the sets $P_m\mathcal{M}_m$ $m = 1,\ldots,M$. There will be but one element from each $P_m\mathcal{M}_m$ as follows from Lemma 1. This set we call $\mathcal{J}_j$ where $j = j_1$ in the symbol (mj_1). Continuing on in this fashion, we construct all sets $\mathcal{J}_j$ and since at least one set $P_m\mathcal{M}_m$ has a designation for the J jobs, we have

Lemma A 2. There are J sets $\mathcal{J}_j$.

We require

(A2) All $\mathcal{J}_j$ and any permutation of the elements of $\mathcal{J}_j$ designated as $Q_j\mathcal{J}_j$ are simply ordered.

It is usual in scheduling to have each object processed on some of the machines. However, we will require

(A3) For all (mj_k), $(m < M)$, $\exists$ an $(m+1\ j_{k'}) \simeq (mj_k)$ attaching a zero time of processing (defined below) to those elements (mj) which represent objects that are not

processed on the m^{th} machine.

In many scheduling problems the determination of a best schedule is predicated on a property of the processing times, although other considerations are sometimes used [4,5,8]. However, in all of the cases treated, the processing time of each object enters into the considerations; therefore, we give

Definition 4. To each $(m_\ell j_k)$ there is assigned a non-negative number $t_{m_\ell j_k}$ which is called the processing time of $(m_\ell j_k)$.

In this paper our considerations will revolve around problems in which we require

(A4) The processing time of each job is characteristic of the job and not the job order.

(A5) An object is processed as soon as possible subject only to the order requirements given by the simply ordered sets $P_m \mathcal{M}_m$ and $Q_j \mathcal{J}_j$.

We can now define a machine shop schedule:

Definition 5. An M x J machine shop schedule is specified by the simply ordered sets $P_m \mathcal{M}_m$ and $Q_j \mathcal{J}_j$ which satisfy (A1-5).

From the definitions, rules and Lemma A2, we easily find

Lemma A 3. For all $P_m \mathcal{M}_m$ and $Q_j \mathcal{J}_j$

(A 1) $$P_m \mathcal{M}_m \cap Q_j \mathcal{J}_j = (mj) .$$

It is usual to give the simply ordered sets $Q_j \mathcal{J}_j$ and ask for the sets $P_m \mathcal{M}_m$ satisfying some condition involving the processing times. At first sight it may be thought that any sets $P_m \mathcal{M}_m$ are possible satisfying (A1-5) when we are given the sets $Q_j \mathcal{J}_j$. However, simple examples dispel this idea. Although the problem of consistency has received a great deal of attention [cf. refs. 1,5,8] we proceed to study the consistency question in our framework.

In order to fix our ideas about the intuitive concept of consistency, we will consider the system made up of the $\{P_m \mathcal{M}_m\}$ and $\{Q_j \mathcal{J}_j\}$, say $\{P_m \mathcal{M}_m\} \otimes \{Q_j \mathcal{J}_j\}$. Although the $\{P_m \mathcal{M}_m\}$ and the $\{Q_j \mathcal{J}_j\}$ are individually simply ordered, the system $\{P_m \mathcal{M}_m\} \otimes \{Q_j \mathcal{J}_j\}$ is not simply ordered.

For consider an element $(mj) \in P_m \mathcal{M}_m$. It is related to all elements in $P_m \mathcal{M}_m$ for $P_m \mathcal{M}_m$ is simply ordered, and (mj) is related to all elements in $Q_j \mathcal{J}_j$ for $Q_j \mathcal{J}_j$ is simply ordered. However, let us consider those $(m_{\ell'} j) \in Q_j \mathcal{J}_j \Rightarrow (m_{\ell'} j) \lesssim (mj)$ and those $(mj_{k'}) \in P_m \mathcal{M}_m \Rightarrow (mj_{k'}) \lesssim (mj)$. These elements, $(m_{\ell'} j)$ and $(mj_{k'})$, are not related because they are in simply ordered sets that are disjoint. For example $(m_{\ell'} j) \in P_{m_{\ell'}} \mathcal{M}_{m_{\ell'}}$ and $(mj_{k'}) \in Q_{j_{k'}} \mathcal{J}_{j_{k'}}$ have only the element $(m_{\ell'} j_{k'})$ in common and this element is not

any one of the elements $(mj_{k'}) \in P_m \mathcal{M}_m \ni (mj_{k'}) \precnsim (mj)$ or $(m_{\ell'}j) \in Q_j \mathcal{J}_j \ni (m_{\ell'}j) \precnsim (mj)$. Hence we have

Theorem A 1. The order relations for a machine shop schedule are partially ordered.

It is the partial ordering of the elements of a machine shop schedule which gives rise to the problem of consistency. The following definitions are useful.

Definition 6. If $(mj) \precnsim (m'j')$ and $(mj) \neq (m'j')$, we write $(mj) \prec (m'j')$. Intuitively the binary relation $\prec$ means "is processed before."

Definition 7. Given the simply ordered $\{Q_j \mathcal{J}_j\}$, the simply ordered $\{P_m \mathcal{M}_m\}$ are consistent if and only if the order relations given by $\{P_m \mathcal{M}_m\} \otimes \{Q_j \mathcal{J}_j\}$ are not inconsistent with the order relations given by the $\{Q_j \mathcal{J}_j\}$.

In Definition 7 we have assumed $\{Q_j \mathcal{J}_j\}$ as given. This is usually the case; the specification of the $\{Q_j \mathcal{J}_j\}$ is the so called technological ordering, and the specification of the $\{P_m \mathcal{M}_m\}$ is the so called scheduled ordering.

The inconsistency arises when a schedule $\{P_m \mathcal{M}_m\} \otimes \{Q_j \mathcal{J}_j\}$ requires an object to be processed both before and after another object. In order to check the consistency of a schedule we consider the strings of elements related via $\prec$ composed of any combination of the following forms:

$$(2.2) \qquad (mj_k) \prec (mj_{k+1}) = (m_\ell j) \prec (m_{\ell+1} j),$$

$$(2.3) \qquad (m_\ell j) \prec (m_{\ell+1} j) = (m'j_{k'}) \prec (m'j_{k'+1}),$$

(2.4) $$(mj_k) \prec (mj_{k+1}) \prec (mj_{k+2}),$$

and

(2.5) $$(m_\ell j) \prec (m_{\ell+1} j) \prec (m_{\ell+2} j).$$

If in any of these strings the same element appears twice, we have the situation that, say,

$$(mj_k) \prec (m_\ell j)$$

and

$$(m_\ell j) \prec (mj_k),$$

where we have eliminated the in-between elements using (A1-2) and Definitions 3 and 6. These relations, usually called "loops" [1], are inconsistent because they require (mj_k) to be processed before and after $(m_\ell j)$. It is sufficient that all possible strings end, i.e. reach an element that is both the last element of some $P_m \mathcal{M}_m$ and $Q_j \mathcal{J}_j$, if we are to have a consistent schedule:

Theorem A 2 (Akers and Friedman). A schedule $\{P_m \mathcal{M}_m\} \otimes \{Q_j \mathcal{J}_j\}$ is consistent if and only if there are no loops in all possible strings composed of any combination of elements of the forms (2.2-5).

Theorem 2 in a different form was given by Akers and Friedman [1] for the purpose of eliminating the inconsistent schedules; Theorem 2 in our form stresses the condition for consistency. The question of a consistent schedule is of

more importance from the point of view of statistical sampling (Monte-Carlo) techniques than the question of elimination of all inconsistent schedules.

Bibliography

[1] Akers, S. B., Jr. and Friedman, J., "A non-numerical approach to production scheduling problems," J. Opers. Res. Am. 3, No. 4, 429-442 (1955).

[2] Birkhoff, G., Lattice Theory, Am. Math. Soc. Coll. Pub. XXV, New York City (1948).

[3] Heller, J., "Combinatorial and statistical aspects of an M x J scheduling problem," AEC Research and Development Report NYO-2540 (1959).

[4] Smith, W. E., "Various Optimizers for Single Stage Production," Management Sciences Research Project, UCLA Res. Report 41, (1955).

[5] Marimont, R. B., "A new method for consistency of precidence matrices," J.A.C.M., Vol. 6, No. 2, 164-171 (1959).

[6] Mitten, L. G., "Sequencing n jobs on two machines with arbitrary time lags," Man. Sci., Vol. 3, No. 3, 293-298 (1959).

[7] Nelson, R. T. and Jackson, J. R., "SWAC Computations for some m x n scheduling problems," J.A.C.M., Vol. 4, No. 4, 438-441 (1957).

[8] Sisson, Roger L., "Methods of sequencing in job shops -- a review," J. Opers. Res. Am., Vol. 7, No. 1, 10-29 (1959).